

SOP for Deploying Flask Application on Azure Web Apps

- Soham Deshmukh

Objective:

This Standard Operating Procedure (SOP) outlines the steps required to deploy a Flask application on Azure Web Apps.

Github Link - <https://github.com/som-d/Python-Flask-App-Deploying-on-Azure-Web-Apps>

Prerequisites:

1. Azure Account

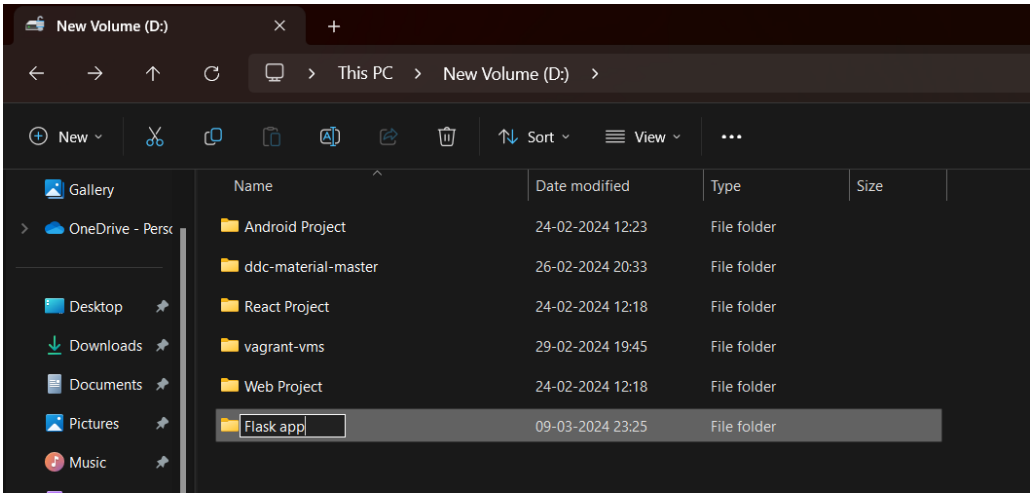
2. Github Account

3. Git: Install Git on your local machine. You can find installation instructions(<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>).

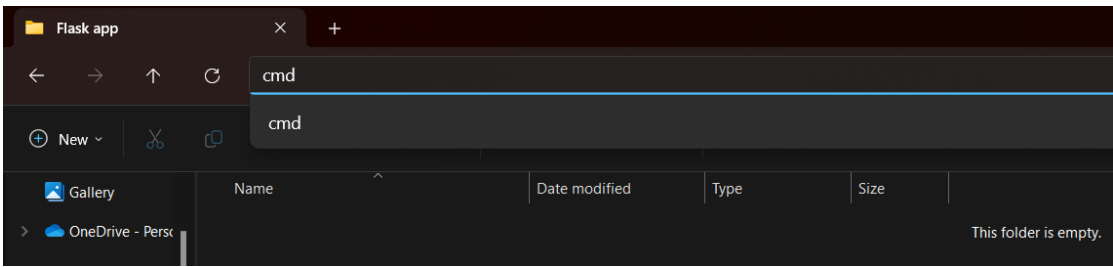
Procedure:

1. Application Setup:

1. Creating Folder for Flask application code:



2. Open the folder and type cmd in search field:



3. Creating an environment for the Flask app:

Command : `python -m venv .`

```
C:\Windows\System32\cmd.e  X  +  v
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

D:\Flask app>python -m venv .|
```

4. Activating the Virtual Environment:

Command : `.\scripts\activate`

Installing Flask : `pip install Flask`

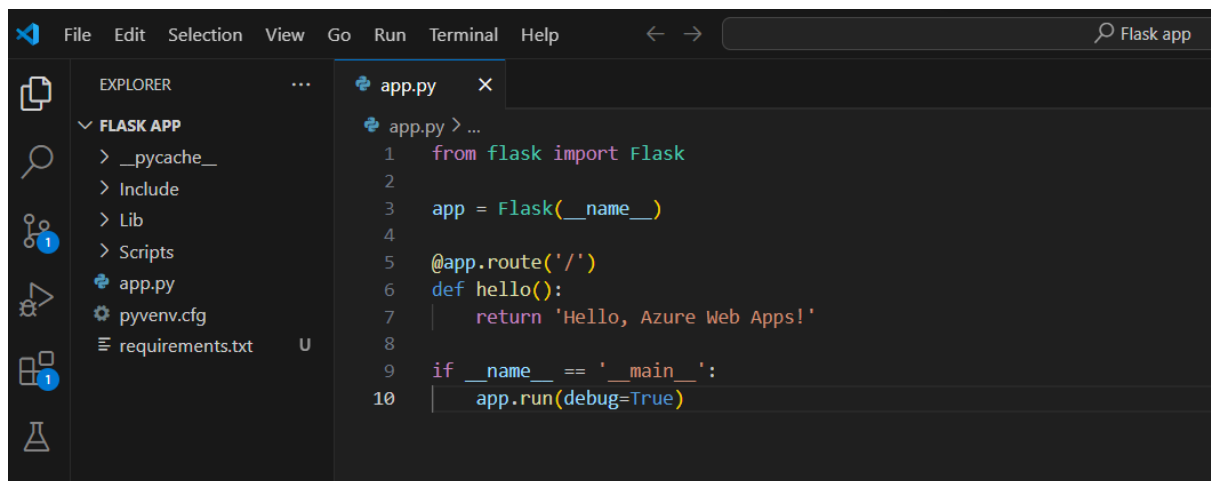
```
D:\Flask app>.\scripts\activate
(Flask app) D:\Flask app>pip install Flask
Collecting Flask
  Obtaining dependency information for Flask from https://files.pythonhosted.org/packages/93/a6/aa98bfe0eb9b8b15d36cdfd03c8ca86a03968a87f27ce224fb4f766acb23/flask-3.0.2-py3-none-any.whl.metadata
  Downloading flask-3.0.2-py3-none-any.whl.metadata (3.6 kB)
Collecting Werkzeug>=3.0.0 (from Flask)
  Obtaining dependency information for Werkzeug>=3.0.0 from https://files.pythonhosted.org/packages/c3/fc/254c3e9b5feb89ff5b9076a23218dafbc99c96ac5941e900b71206e6313b/werkzeug-3.0.1-py3-none-any.whl.metadata
  Using cached werkzeug-3.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting Jinja2>=3.1.2 (from Flask)
  Obtaining dependency information for Jinja2>=3.1.2 from https://files.pythonhosted.org/packages/30/6d/6de6be2d02603ab56e72997708809e8a5b0fbfee080735109b40a3564843/jinja2-3.1.3-py3-none-any.whl.metadata
  Downloading Jinja2-3.1.3-py3-none-any.whl.metadata (3.3 kB)
Collecting itsdangerous>=2.1.2 (from Flask)
  Obtaining dependency information for itsdangerous>=2.1.2 from https://files.pythonhosted.org/packages/68/5f/447e04e828f47465eeab35b5d408b7ebaaee207f48b7136c5a7267a30ae/itsdangerous-2.1.2-py3-none-any.whl.metadata
  Downloading itsdangerous-2.1.2-py3-none-any.whl.metadata (2.9 kB)
Collecting click>=8.1.3 (from Flask)
  Obtaining dependency information for click>=8.1.3 from https://files.pythonhosted.org/packages/00/2e/d53fa4befbf2cfa71
```

5. Creating Requirements.txt :

Command :	<code>pip3 freeze > requirements.txt</code>	# Python3
	<code>pip freeze > requirements.txt</code>	# Python2

Note – in requirement.txt comment/delete: `pywin32==306`

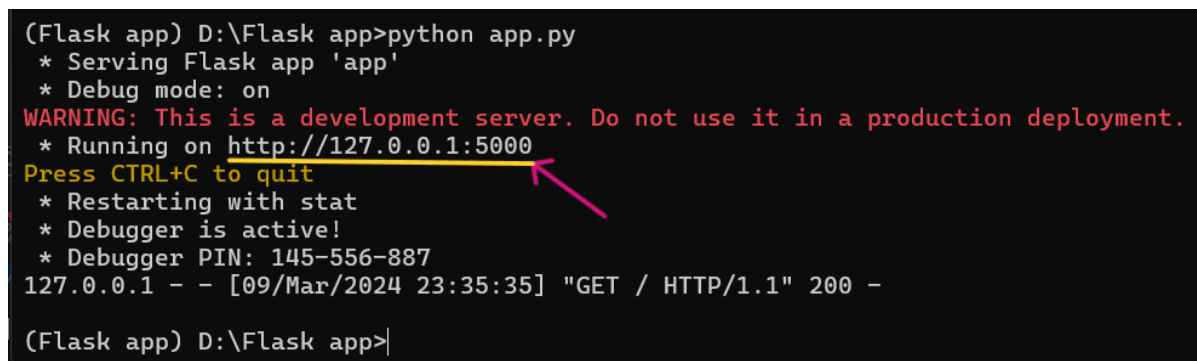
6. Creating app.py:

A screenshot of the Visual Studio Code editor. The Explorer sidebar on the left shows a project named 'FLASK APP' with files like '__pycache__', 'Include', 'Lib', 'Scripts', 'app.py', 'pyvenv.cfg', and 'requirements.txt'. The main editor window shows the content of 'app.py', which contains Python code for a Flask application. The code imports Flask, creates an app instance, defines a route for '/', and runs the app in debug mode.

```
app.py > ...
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def hello():
7      return 'Hello, Azure Web Apps!'
8
9  if __name__ == '__main__':
10     app.run(debug=True)
```

7. Starting the server:

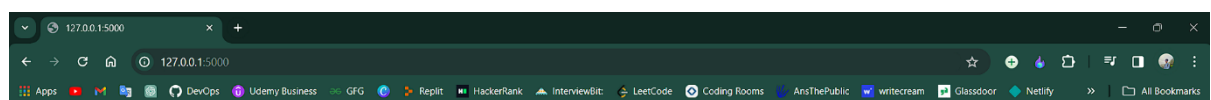
Command : `python app.py`

A screenshot of a terminal window showing the output of running 'python app.py'. The output indicates that the Flask app is serving on http://127.0.0.1:5000. A pink arrow points to the URL. The terminal also shows a warning about development servers and a successful GET request.

```
(Flask app) D:\Flask app>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 145-556-887
127.0.0.1 - - [09/Mar/2024 23:35:35] "GET / HTTP/1.1" 200 -
(Flask app) D:\Flask app>
```

- Copy the URL and past in browser to run Web App.

8. Flask app loaded on local server:

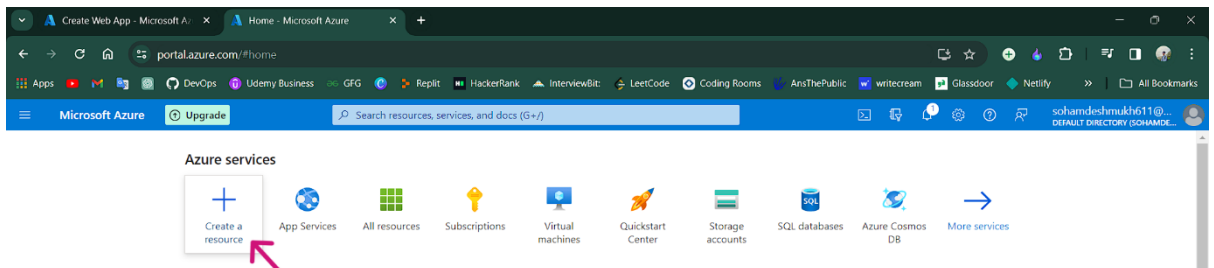


Hello, Azure Web Apps!

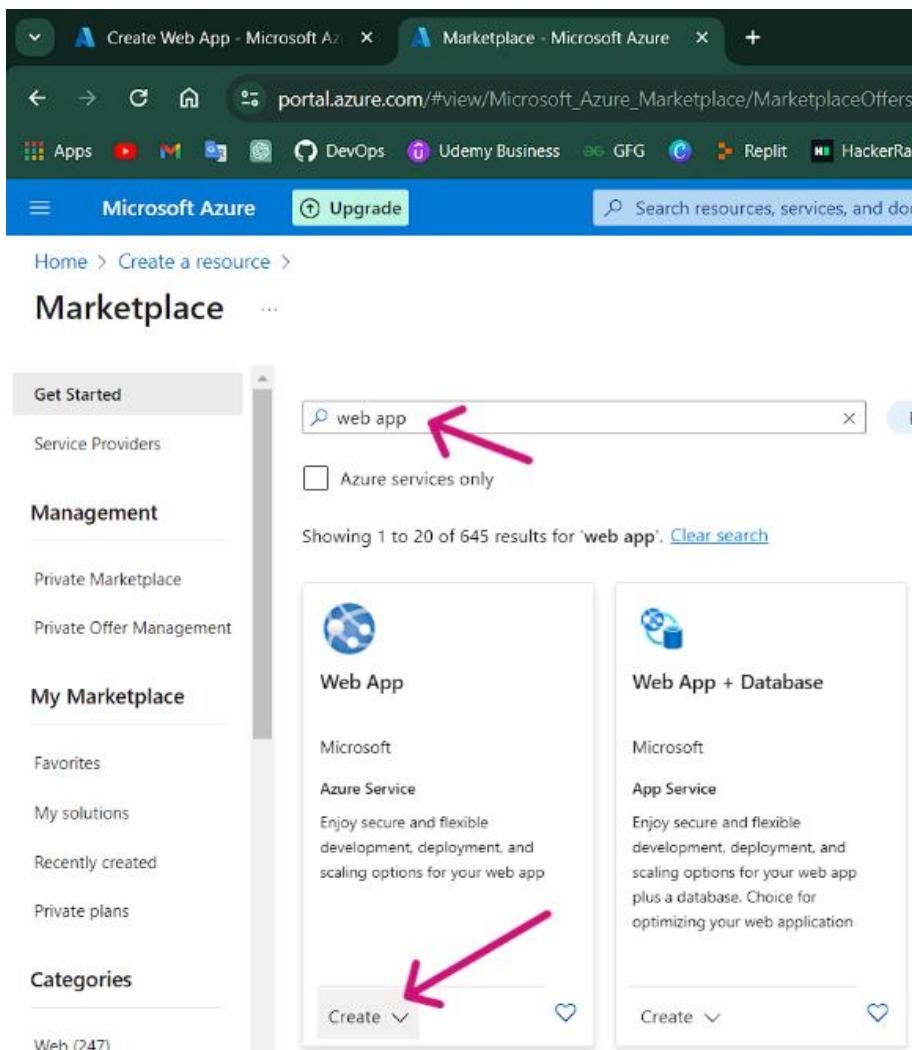
2. Create Azure Web App:

Create a new Azure Web App:

1. Login to Azure and click Create a resource:



2. Search for Web App and click Create:



3. Creating Resource group:

- Click Create New and name the resource group.

portal.azure.com/#create/Microsoft.WebSite

Microsoft Azure

Home > Create a resource > Marketplace >

Create Web App

Basics Database Deployment Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource Group * [Create new](#)

A resource group is a container that holds related resources for an Azure solution.

Name *

Instance Details

Name *

Publish *

Runtime stack *

4. Fill in all required fields and click Create:

The screenshot shows the Azure Portal interface for creating a new Web App. The browser address bar shows the URL `portal.azure.com/#create/Microsoft.WebSite`. The page title is "Create Web App".

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Free Trial ▼

Resource Group * ⓘ Flask_app ▼
[Create new](#)

Instance Details

Name * Flask-App-devops ✓
.azurewebsites.net

Publish * ☒ Code ☐ Docker Container ☐ Static Web App

Runtime stack * Python 3.11 ▼

Operating System * ☒ Linux ☐ Windows

Region * East US ▼

Review + create < Previous Next : Database >

Not finding your App Service Plan? Try a different region or select your App Service Environment.

5. After creating the web app, click Deploy Center:

The screenshot shows the Microsoft Azure portal interface for a web app named 'Flask-App-devops'. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment, Deployment slots, Deployment Center, Settings, Environment variables, and Configuration. The main content area displays the 'Essentials' section with details about the resource group, status, location, subscription, and tags. Below this, the 'Properties' tab is active, showing the 'Web app' section with fields for Name, Publishing model, and Runtime Stack. A red arrow points to the 'Deployment Center' link in the right-hand section of the page.

6. Fill in all required fields and click Save:

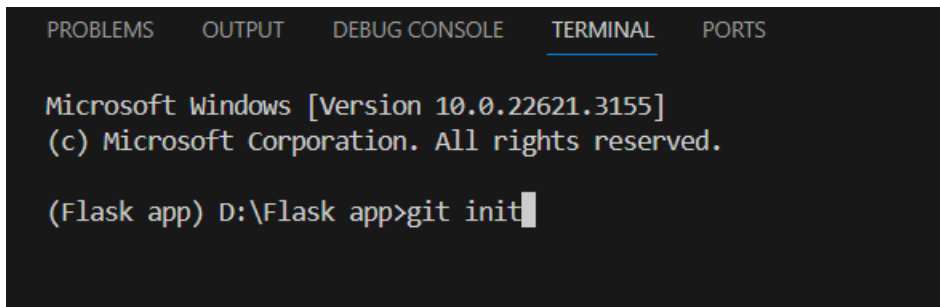
The screenshot shows the 'Flask-App-devops | Deployment Center' configuration page in the Microsoft Azure portal. The left sidebar is the same as in the previous screenshot. The main content area displays the 'Deployment Center' configuration form. At the top, there is a 'Save' button highlighted with a red arrow. Below the 'Save' button, there is a notification banner stating 'You're now in the production slot, which is not recommended for setting up CI/CD. Learn more'. The form includes a 'Source' dropdown menu set to 'GitHub', a 'Signed in as' field showing 'som-d', and fields for 'Organization', 'Repository', and 'Branch', all set to 'som-d', 'Flask-app', and 'master' respectively. A 'Save' button is located at the bottom right of the form.

3. Azure Web App Deployment:

Deploy the application to the Azure Web App using Git:

1. Creating an empty Git repository

Command : git init



```
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

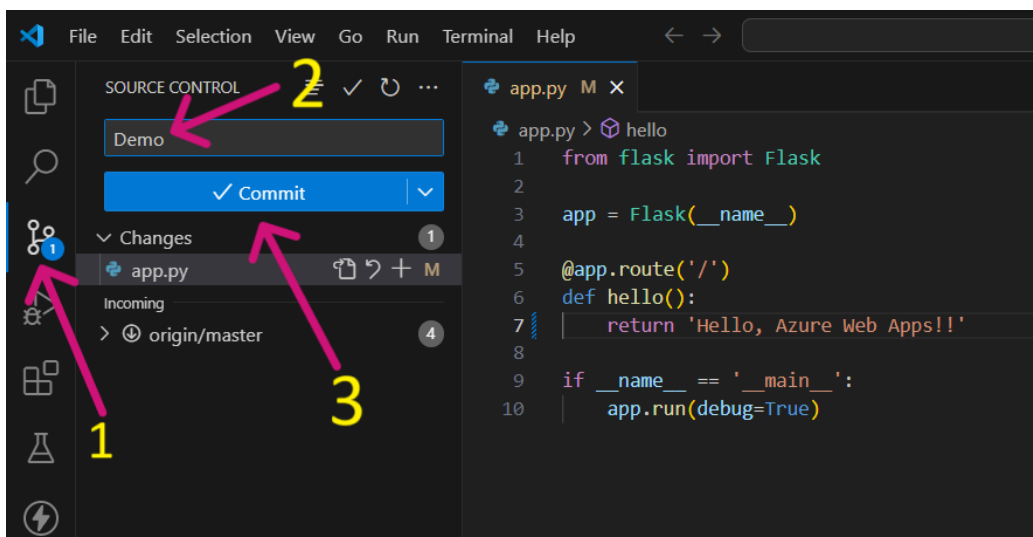
(Flask app) D:\Flask app>git init
```

2. Commit by using GUI:

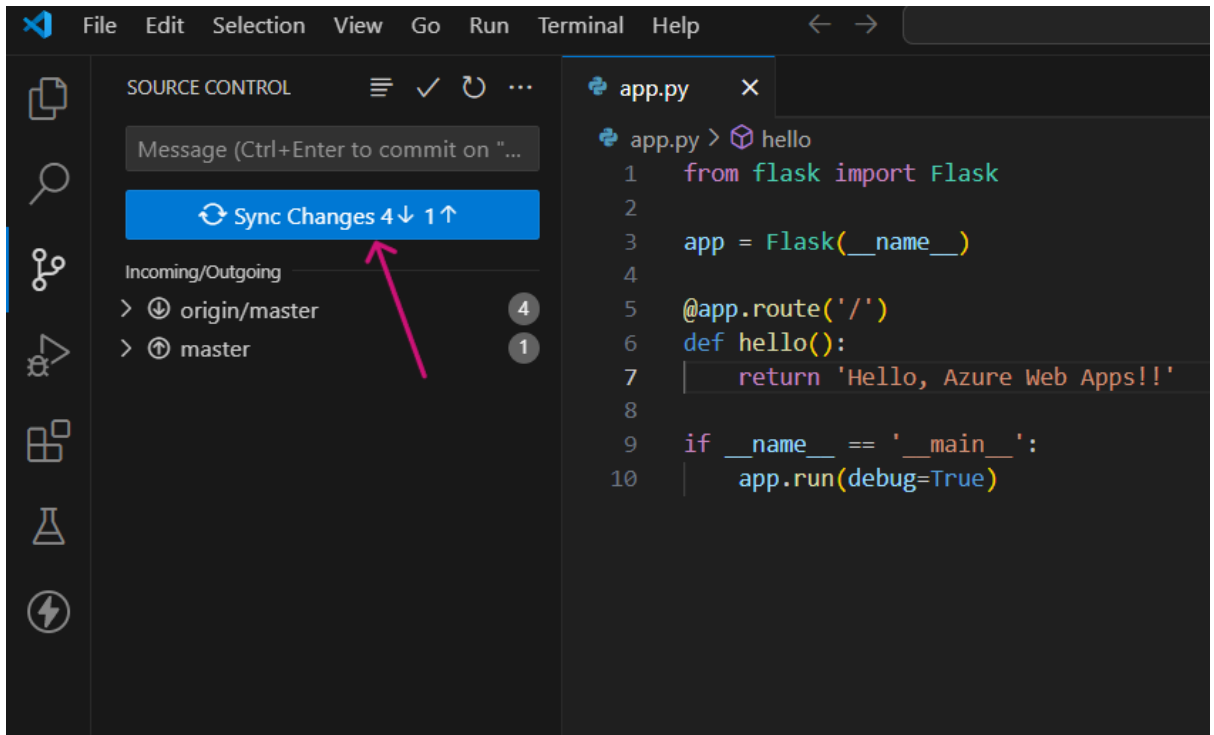
Arrow 1: Click on Source Control.

Arrow 2: Add a comment.

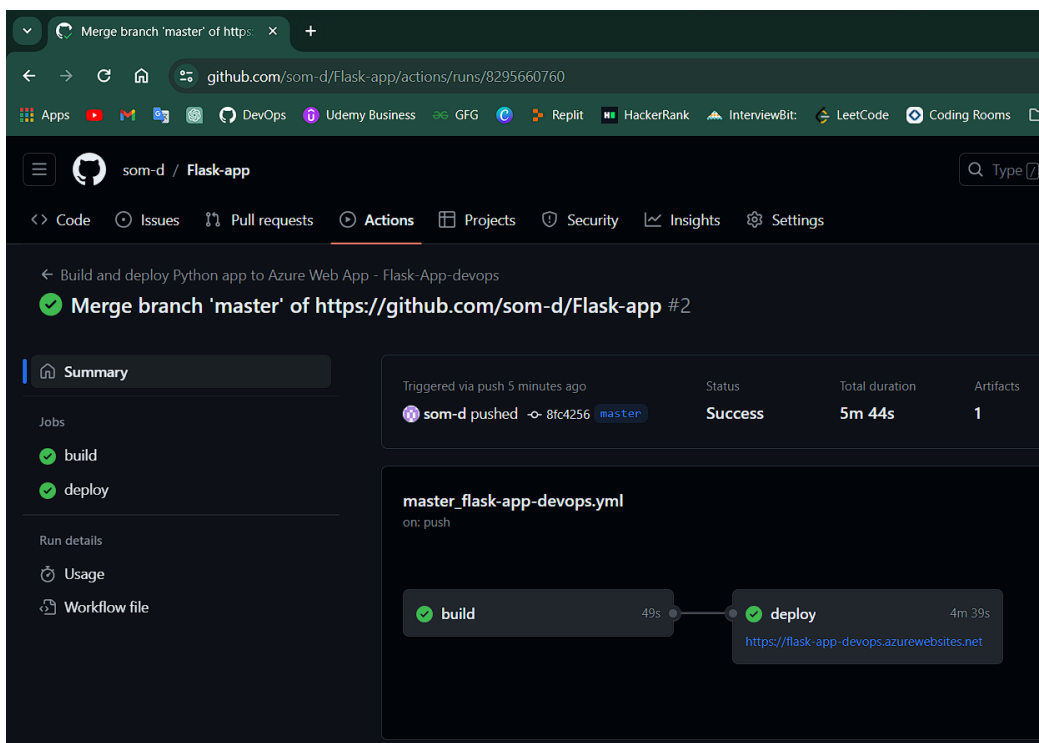
Arrow 3: Click Commit.



3. After that click Sync Changes:

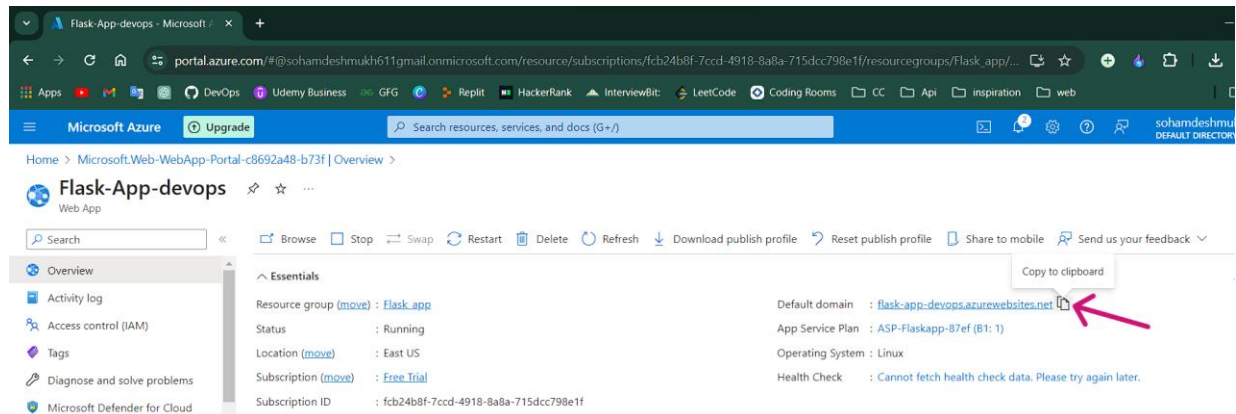


4. Go to your Git repository and click on Actions to see all your workflows:

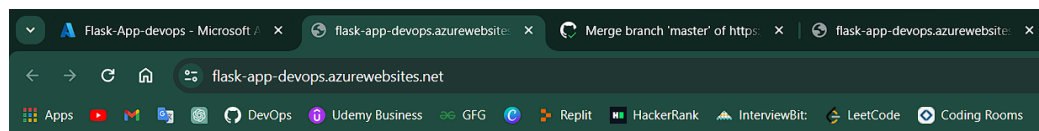


4. Testing:

1. Once the deployment is complete, access the deployed application using the URL provided by Azure Web Apps.



2. You should receive a response "Hello, Azure Web Apps!" indicating that the application is running successfully.



Hello, Azure Web Apps!!

Contributors:

Soham Deshmukh - <https://github.com/som-d>

Contact Information:

For any questions or issues regarding this SOP, please contact: sohamdeshmukh611@gmail.com